UDC 004.89
https://doi.org/10.31891/csit-2023-1-1

Nataliya BOYKO, Roman KOVALCHUK
Lviv Polytechnic National University

DATA UPDATE ALGORITHMS IN THE MACHINE LEARNING SYSTEM

This paper analyzes methods for operationalizing anomaly detection, data drift detection, as a data validation step in a machine learning system. A pipeline is a set of data processing elements connected in series, where the output of one element is the input of the next. MLOps is a set of practices aimed at reliable and efficient deployment and support of machine learning models in the real world. We proposed a solution with technologies mentioned in the theoretical paper [1] for operationalizing the Data QC pipeline. Also, we propose to build a Data QC pipeline based on MLFlow, a machine learning cycle manager. We chose MLFlow as a skeleton for building our pipelines. The choice springs from the specifics of the task, problems and the need for ready-made solutions to meet our requirements. Specific explanations are mentioned in the paper [1] both for Data Drift and Data QC pipelines. To construct either Data QC or Data Drift pipeline, we need to wrap the defined solution, divided into steps to the MLFlow. The latter will register all artifacts, metrics and parameters. An artifact in a machine learning system is a result of a process in a pipeline. For example, it could be a trained model, an Excel file, or a feature importance image. The paper considers the following stages of the Data QC pipeline: filtering, anomaly detection, reporting, validation, and comparison of new data with historical. The Data Drift detection pipeline. The Data QC and Data Drift detection pipelines are necessary for data validation and processing in the current machine learning life cycle. The task of the Data QC pipeline is to automate the evaluation and validation of new data. The task is especially important for Time-Series systems in real-time. In this paper, we researched the formation of interactive quality reports, and the anomaly and data drift detection approaches for the Time-Series system. We analyzed approaches to implementing such MLOps architecture with data validation step described with Data QC and Data Drift pipelines.

Keywords: Data Drift, Data QC, Anomaly Detection, MLOps, Data Validation, Machine Learning, Time-Series.

.

Наталія БОЙКО, Роман КОВАЛЬЧУК
Національний університет «Львівська політехніка»

АЛГОРИТМИ ОНОВЛЕННЯ ДАНИХ В СИСТЕМІ МАШИННОГО НАВЧАННЯ

У цій роботі було виконано аналіз методів для операціоналізації пошуку аномалій, виявлення дрифту даних та самого DataQC пайплайну як такого. Проаналізовані підходи до аналізу операціоналізації пайплайну та до операціоналізації виявлення дрифту даних. Виявлення аномалій допомагає нам оцінити чистоту і якість наших даних. Важливо, щоб у моделі не було аномальних викидів, оскільки вони заплутують модель. Також важливо мати послідовні дані без змін у розподілі ознак. Було запропоновано рішення з вибраними технологіями для операціоналізації DataQC пайплайну, визначено наступні кроки для подальшого дослідження. Запропоновано для побудови заданого DataQC пайплайну використати та обґрунтовати власне рішення для пошуку аномалій та виявлення дрифту даних через специфіку задачі, проблеми та відстуності готових рішень які б задовольняли наші вимоги. В роботі розглядаються етапи операціоналізація вищезгаданого пайплайну, який виконує етапи: фільтрування, пошуку аномалій, звітування, валідації, та порівняння нових даних з історичними, для існуючої у системі моделі машинного навчання. Описується складність задачі операціоналізації у реальному світі, яка полягає у постійному оновленні даних, необхідності їх опрацювання та подальшому застосуванні у системі машинного навчання. Також доводиться користь від пайплайну, який б автоматично опрацьовував нові дані. В роботі досліджується проблематика, яку слід розглядати як Time-Series проблему, то при формуванні інтерактивних звітів, перевірці даних на валідність, наявність та пошук викидів, аномалій. Це рішення дозволить нам візуалізувати всі кроки, які виконує конвеєр валідації даних, що дасть змогу іншим розробникам переглянути результат його роботи, не знаючи нюансів його реалізації та не витрачаючи зайвого часу. Також пропонована архітектура MLOps дозволяє відстежувати зміни трендів даних та гарантувати, що модель збереже свою прогностичну ефективність з часом.

Ключові слова: дрифт даних, пайплан, аномалії, операціоналізація, препроцесинг, машинне навчання.

Introduction

The complexity of building solid MLOps architecture in the real world is constantly updating, processing data, model monitoring, and the need for further use in the machine learning system. The benefits of an architecture that automatically processes new data are undeniable. The cleaner our data is, the easier it is for the machine learning algorithm to work with it, and the more predictable the result will be. Sticking to the MLOps principles ensures quality work for all its users: Data Scientists, Software Engineers, and DevOps.

This work aims to create a Data Validation step in our ML system by introducing Data QC and Data Validation pipelines. They are a wrapper of the ready-to-go theoretical solution presented in the previous work. In order to wrap a Python script into several separate pipeline blocks that perform specific jobs, such as anomaly detection, data filtering, or report generation, we got to use a lifecycle manager like MLFlow [6].

With MLFlow, we can record metrics from each experiment through a visual interface, compare its parameters, and evaluate its effectiveness. The mentioned Data QC pipeline consists of the following steps:

1. Loading data from the database.
2. Filtering and preprocessing of data.
3. Search for anomalies in the data.
4. Finding the difference between new and past anomalies.

5. Generating an interactive report on new data.
6. Checking new data on Data Drift.
7. Uploading pre-cleaned data to the database.
8. Logging parameters, metrics, and results of the pipeline execution

We used Pandas and PostgreSQL for data loading and processing. For interactivity of filters - integration with Microsoft Sharepoint. For anomaly detection, machine learning, statistical methods and their combination. For Drift Detection, statistical methods based on testing the null hypothesis of equality of two distributions and rule-based methods. For report generation - Jupyter Notebooks and Holoviz Panel. For logging artifacts, reports, parameters and metrics - MlFlow Constants. For organizing the pipeline - MlFlow Runs. Also, the most important point is that this solution should be On-Premise, that is, work not on the cloud but on a dedicated server of the company.

The decision to place the service on a dedicated server is due to the company's security requirements, which is due to the high cost of confidential data. The project's dataset covers most or all of the employee's actions in the company, his reporting, salary changes, managers' feedback and work history. The risk of such data leakage into the public domain is highly undesirable for the company due to reputational losses, which correspond to monetary losses and data confidentiality issues. Also, data leakage is undesirable due to possible legal problems, leading to reputational and monetary costs.

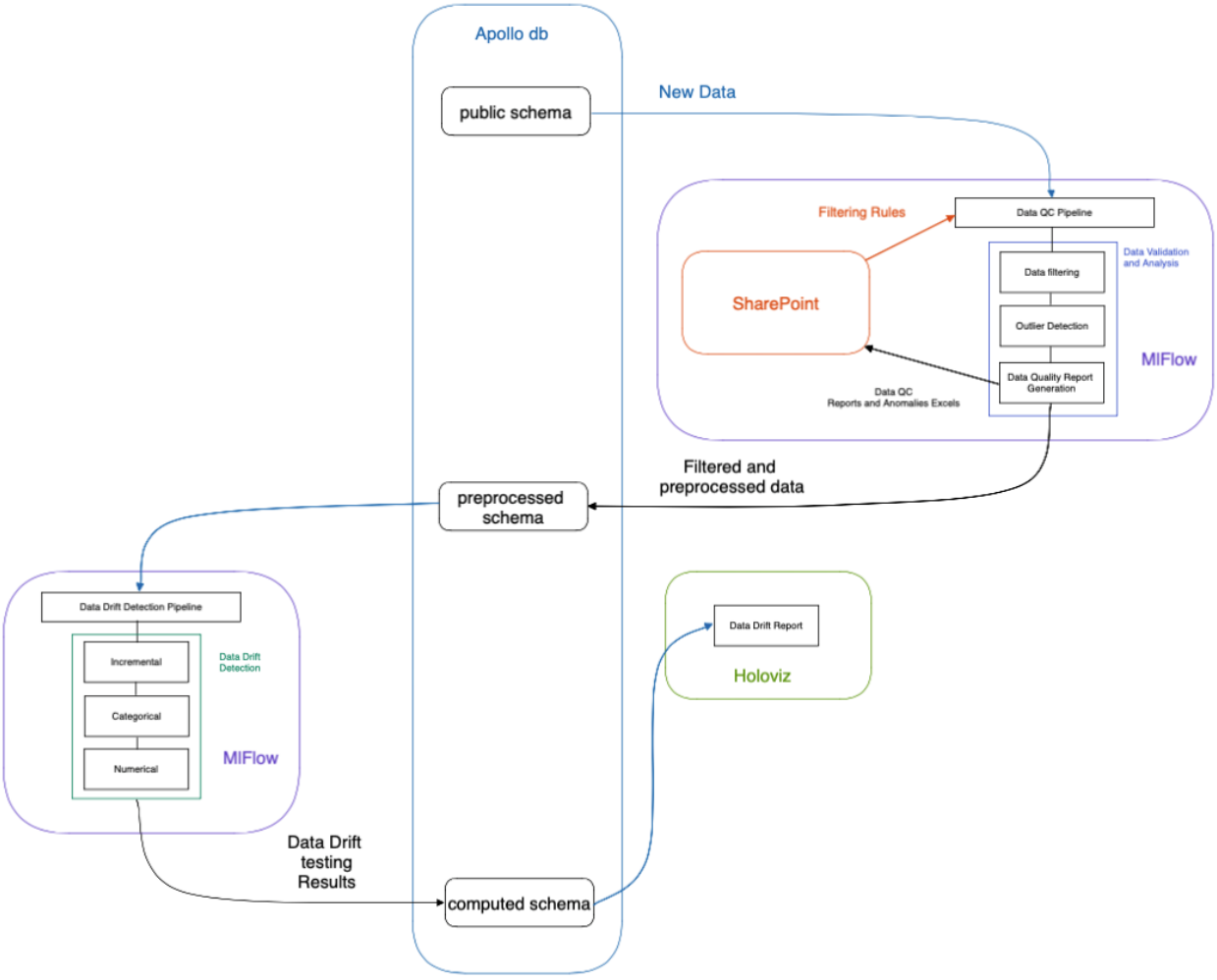

Fig. 1 An example of the architecture of the operationalized solution of DataQC and Data Drift pipelines

According to Fig. 1, a visual representation of the solution's MLOps architecture, we can see that Data Validation is split into DataQC and Data Drift pipelines. Each loads intermediate results into the database. This decision, in addition to architectural rules, for example, separation and isolation of individual tasks into separate pipelines, is also due to practicality. The execution time of the Data Drift pipeline is 7 hours on a dedicated server. In contrast, the Data QC pipeline takes only 15 minutes.

The Data Drift Pipeline is distributed into computing tests and visualizing the result. Firstly, we detect drift in the Data Drift pipeline and write the results to the computed schema of our database. Then, if visualization is necessary, run a Python script that will pull up the necessary data without recalculation. Hosting of visualization page is necessary due to the specifics of working with the Holoviz Panel [3] package.

Although we abstracted ourselves from the model in this work, let us consider it for completeness. We chose a rather complex LightGBM heuristic machine learning model based on decision trees with gradient boosting. The dataset limits the use of transformers or neural networks. Although, with a larger dataset, it could be more efficient in identifying dismissed workers. The following data sources are available:

1. Personal data, e.g. gender, year of birth
2. Status of the employee, e.g. whether he/she is in reserve or dismissed
3. The employee's position, management level and job profile
4. The employee's languages
5. The employee's compensation, bonus history and scheduled salary reviews
6. The customer of the project
7. Project on which the employee is working
8. Certifications that the employee has passed
9. Aggregated Peakon score of the employee on the company and his team
10. Information about the employee's professional review
11. Feedback on the employee from his manager
12. The employee's manager

We can see the depicted model in Fig. 2. Tabular structured data ready for processing is coloured in green. Moreover, purple indicates unstructured text that will be transformed into structured data. This transformation can be performed by the BERT classifier, pre-trained on the GoEmotions [5] dataset, which has a similar specificity to ours, evaluating texts by emotions. Then from these emotions, we can extract the sentiment of the response, for example, whether it is positive or negative. Then, we combine the obtained structured data into single dataset.

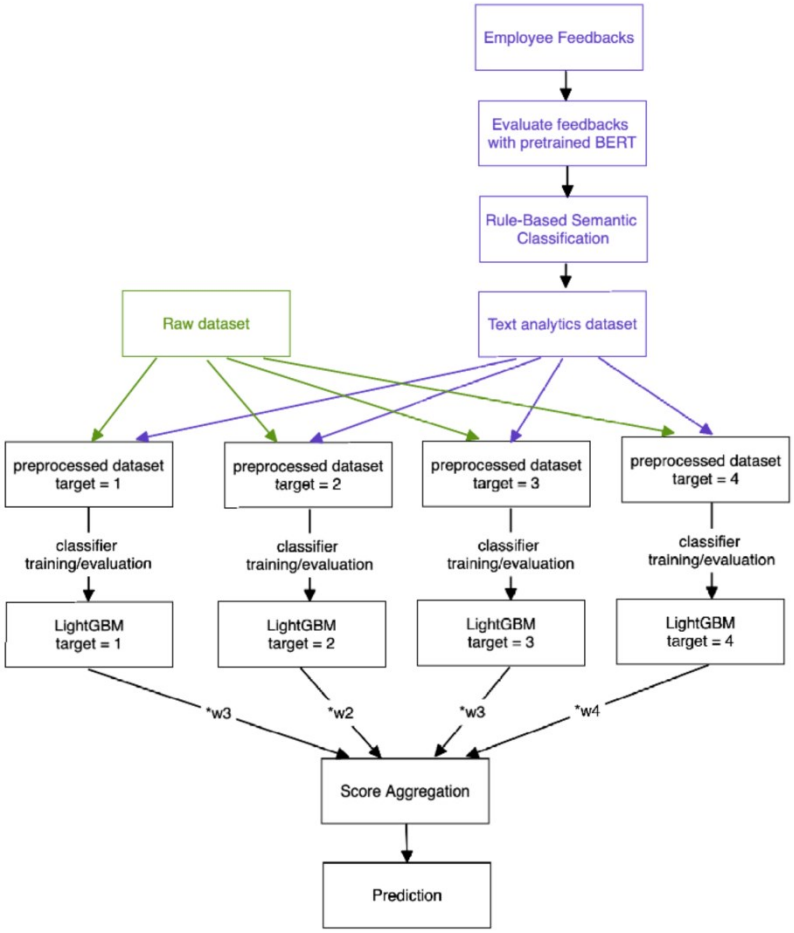

Fig. 2 Example of ensemble architecture of LightGBM models

The "target" is responsible for the hyperparameter of the same name, which corresponds to how much we extend the employee's dismissal into history. How many months in advance do we want to calculate whether the employee will dismiss? For example, with target = 1, we estimate the dismissal in the next month, and with target = 2, in the next two months inclusive.

This work aims to wrap the validation and evaluation of data into pipelines, which in this architecture, in Fig. 2, outputs structured data marked with a green block.

The object of research is a system for predicting the probability of dismissal of a particular employee in a

company after a specified time. A large number of possible independent variables characterizes the system. For example, the model uses about 200 features, some of which are generated.

Since dismissal prediction is a Time-Series task, it is necessary to pay attention to trends and seasonality. This remark also applies to generating interactive reports, verifying data for validity, and anomaly detection.

For instance, there is a trend towards increasing salaries in certain profiles. Hence, the distribution is not stale and is constantly shifting right.

When applying statistical methods, we have to adjust to this error. For instance, assuming Architectors got a plus 10% of their salaries over the next four months, the salary distribution shape for Architectors remains, and we consider this as not an anomaly. Hence, drift and anomaly detection methods should not consider this behaviour anomalous. As an example of an application - search for anomalies in the salary column.

Similarly, we should consider this nuance when checking a column for Data Drift. For instance, the salary distribution for Architectors has dropped by 5% over the last four months. Moreover, the historical data shows us a shifted but identical distribution. Then, the Data Drift should not be detected. Because the salary distributions, except for the conditional mean, are identical. So, we are only interested in detecting the change in the shape of the distribution.

Analysis of recent sources

Analyzing the previous article [7], which describes the theoretical methods for building the data validation step in MLOps architecture, let us briefly recall its components and requirements. So, among the components of the solution of the previous article:

1. Loading data from the database
2. Data filtering and preprocessing
3. Anomaly detection
4. Monitoring the difference between new and past anomalies
5. Generating an interactive report on new data
6. Checking Data Drift on new data
7. Uploading cleaned data to the database
8. Logging parameters, metrics, and results of the pipeline execution.

We have decided to split mentioned in paper [7] data pipeline into two separate ones. Respectively, DataQC and Data Drift Detection pipelines, according to the architecture in Fig. 1 above.

Although the article mentions public packagessuch as Evidently AI [1], we constructed our solution to meet all the requirements. Remembering that we need to wrap these two pipelines in a common architecture, we need to containerize them. So, we will need to split our code in the pipeline into some blocks and steps and use MlFlow to organize logging and tracking experiments and parameters. Tracking is monitoring the results of the execution of a pipeline or a certain job. A pipeline job is the same as a pipeline step. For example, let us single out the Data QC steps of the pipeline from the list above:

1. Loading data from the database
2. Data filtering and preprocessing
3. Anomaly detection
4. Finding the difference between new and past anomalies
5. Generating an interactive report of new data validation
6. Uploading cleaned data to the database
7. Logging of parameters, metrics and results of the pipeline execution

In the list above, for example, from step 3, described in the article [7] of methods and parameters, we form an abstraction in the form of a base and several child classes, one for each implementation of the anomaly detection method, if necessary. Furthermore, one more class would perform all the necessary preprocessing and calls to initiate the work of the previous one, acting as a wrapper. Does building a well-structured OOP code affect the quality of the model? Directly - no, we can copy the linear script of the program, paste MLFlow calls to the API, and finish. However, we will immediately face several anti-patterns in the MLOps world. The first of them is abstraction debt, plain-old-data type smell and glue-code. These anti-patterns are described in detail in the Google study [8], which describes the importance of MLOps architecture in the modern Data Science world. In short, we want to avoid duplicating code but reusing the same logic as much as possible to guarantee the experiment's repeatability. An experiment returning a good model is only possible if we know which parameters to reuse, how to improve the model, or how to debug it. One of the reasons for not reproducible code is just duplicates, where some small error happens. It is considered a good practice to have a well-structured OOP code that is easy to maintain.

After implementing Data QC and Data Drift pipelines with MLFlow, we should containerize these two pipelines using Docker. And since both perform ETL (Extract, Transform, Load), the result of the pipeline execution is always loaded into a common database. By the way, the central idea of our architecture is a shared database with several schemas that display the intermediate results of some pipelines or experiments, see Fig. 1, above. By the way, this necessity arises from banal convenience. For example, we will run the Data QC pipeline, possibly several times with a new batch of data, updated once a month. The result of the Data QC pipeline affects the quality of the model,

as the amount and value of data changes, due to the search and correction of anomalies. We can experiment with the parameters of certain methods mentioned in the article [7]. However, the Data Drift pipeline does not directly affect the model but the drift results. Therefore, there is no need to run it more often than the data is updated. Actually, it was invented for this purpose. For the initial validation of data, when they are updated.

In this article, we will elaborate on the application of the Data Drift pipeline in much more detail than its construction and tests described in the paper [7]. It is much more often used in a full-fledged MLOps architecture. It is especially often when compering with, for example, anomaly detection, which can be found in other articles as a step of a training pipeline or script without recording intermediate results.

Presenting main material

To study the construction of the DataQC pipeline, we identified the necessary features for which we needed to make validation.

Almost every column presented a slightly different approach to solving the problem. For example, we need to apply ANOVA and Percentile filtering to search for anomalies in the multimodal column WageGross. For other numerical columns, we can use the Median + IQR method, for which the only condition is the normality of the distribution. Having evaluated these data types, we formed a list of the necessary anomaly search functions for each of the columns. It is essential to record all the metrics we have obtained. For example, how many employees have null-salary, how many salary values we have filled with past values, the number of anomalies by column, and the parameters we have used. In addition to banal convenience, notation and reproducibility of experiments are key features in building a high-quality artificial intelligence system. For example, after launching the MlFlow client and building a Data QC pipeline, we can observe the parameters with which we ran a certain experiment and what metrics it gave. Anomaly detection or data validation is not a supervised learning task. So, we cannot immediately assess the quality of our Data QC pipeline and how well we handle anomalies. However, we can assess the quality of anomaly processing and data visualization and, ultimately, by assessing the model itself, which is already a supervised-learning task.

In the study on building a given Data QC pipeline, we used and justified our solution for finding anomalies and detecting data drift due to the specifics of the task. Therefore, we should consider the nuances of the MLOps part in more detail.

Among the main requirements: are autonomy, the flexibility of visualization, the flexibility of modifying the logic of the anomaly detection method, and resistance to shifts in distributions.

First, let us consider the requirements for the anomaly search part of the pipeline:

- Search for anomalies by specified columns: APM, WageGross, OnSite, MonthOnPosition and VacancyHistory table.
- Deleting or filling these anomalies from the dataframe.
- Writing these anomalies to Excel files and uploading them to Microsoft SharePoint for automatic monitoring and their elimination at the level of data owners.
- Interactive visualization of the found anomalies.

Since we already have the implementation of the anomaly detection method, the task of operationalizing this step will be to visualize and monitor the result. Considering that we have chosen Bokeh, HVPlot and HoloViews to visualize the found anomalies, we need an interface to display these plots. In this case, two options are available. The first is to group the graph output functions into Jupyter Notebook, run it and convert it to HTML (since the graphs are interactive). And the second is to group the graph output functions into Holoviz Panel. This visualization hosting tool integrates well with the HvPlot platform, HoloViews.

Holoviz Panel is harder to implement as we deploy and describe a service with visualizations. However, the finished service is easy to use. It is always available and is very flexible to modify as we build it.

Converting Jupyter Notebook into an HTML page is much easier but more limited in terms of interactivity, as all the code will be translated into JavaScript. However, this option will display the same charts without interactivity between them, but only in a separate chart.

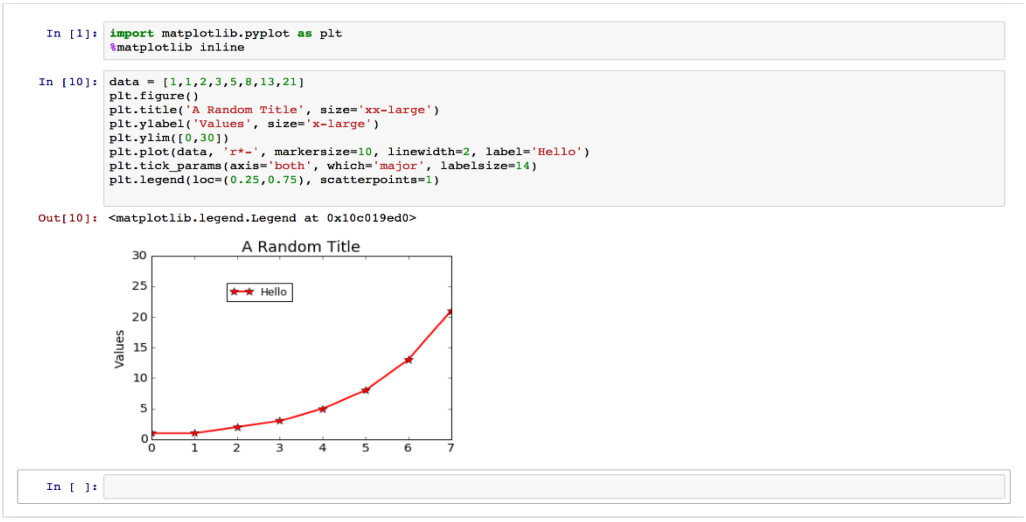

Fig 3. Example of report visualization using Jupyter Notebooks

For the task of visualizing anomalies, Jupyter Notebook is enough for us, and there is no need to describe the page with Holoviz Panel, although the latter has its advantages mentioned above.

After generating the report, the next step is to convert it to HTML and upload it to Microsoft SharePoint for access by stakeholders and other team members.

Analysis of approaches to bring data drift detection into our MLOps architecture

With data drift detection, we also decided to have our solution. However, we should also mention the cases when we would not use a ready-made package for drift detection. So, let us compare ready-made solutions for data drift detection.

From Fig. 4, consider TensorFlow Data Validation (TFDV) [2]. This tool is an addition to the TensorFlow package and neural networks, part of the ML infrastructure supported by Google - TensorFlow Extended (TFX). This option is unsuitable for us because our model is not a neural network but a LightGBM model. The Whylabs tool does not support non-cloud solutions and, therefore, does not suit us since our MLOps architecture is on-premise. Great Expectations package is also unsuitable because it does not support data drift detection. Evidently AI, already mentioned in the previous paper, was not chosen because it does not support the modification of reports and is limited in modifying testing methods to detect data drift. Due to the above reasons, we have chosen our solution again.

Fig. 4 Comparison of ready-made tools for data drift detection

Since data drift is a more ordinary task, the detection will determine whether the training will occur. We need a visualization tool to interactively compare the graphs of many features to identify where exactly the data drift occurred or what data is invalid. On the contrary, data visualization is more than a fait accompli, which we have to show and not analyze in detail.

Given the previous paragraph, it would be logical to choose the above-mentioned Holoviz Panel tool. Because with it, we can create an interactive page for each available data type, numeric, categorical and incremental, to check the data validity and the presence of data drift.

Analysis of machine learning lifecycle managers

Taking into account the simplicity and linearity of our pipeline, among the available ready-made solutions, for example, TensorFlow Extended (TFX), which we reject due to the lack of TensorFlow and neural networks in our solution. Amazon Sagemaker [4] and similar cloud solutions, which we also reject due to on-premise, we remain on a simpler solution - MlFlow.

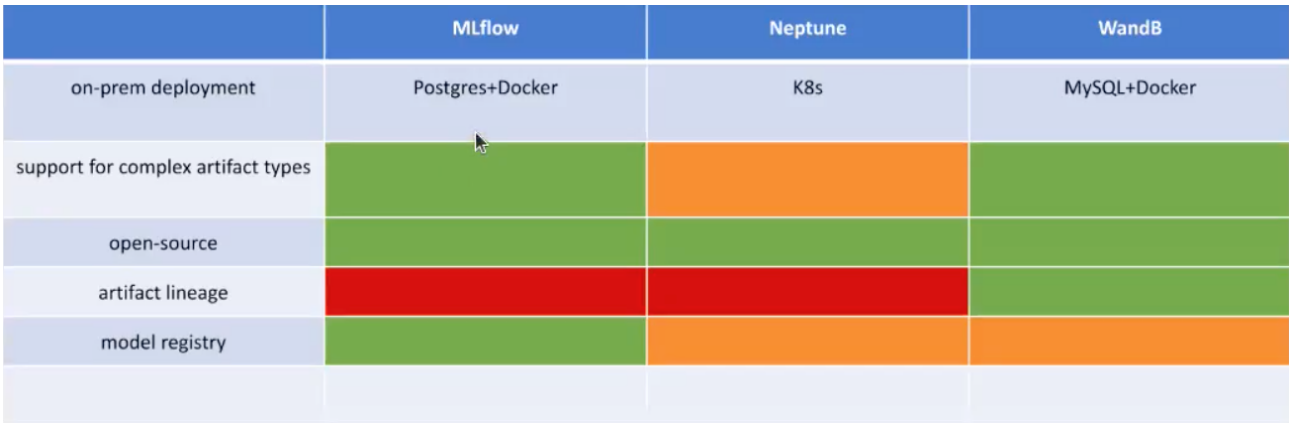

Fig. 5. Comparison of machine learning-lifecycle manaers

We compared machine learning lifecycle managers in Fig. 5.

Note that we chose Docker and Docker-Compose as a tool for deployment and containerization. We do not plan to deploy our solution to any of the clouds since our solution is an On-Premise solution and should be run on our dedicated server. However, when it comes to cloud solutions, Kubernetes (K8s) and Docker are the favourites because of their easy integration and support. It is much more difficult to raise, configure and maintain a K8s cluster on our server than on the cloud. We are responsible for load balancing and expanding the machine's capacity. Moreover, cloud providers usually take this role on themselves.

The next factor for choosing simpler containerization with Docker is that it is much easier to work with and configure. At the same time, Kubernetes focuses on a heavier infrastructure, which includes CI/CD integration. We have this opportunity limited due to the company's security policy.

Conclusions

This paper analyzes methods for efficient deployment and the use of anomaly detection and data drift detection methods in the real world.

We proposed a solution with selected technologies for operationalizing the data validation step of a machine learning system. We identified the following steps for further research, namely:

1. To implement and document the Data QC architecture of the pipeline as a step of data validation before data processing.
2. Operationalize anomaly detection and data drift detection steps using Jupyter Notebook, MlFlow Tracking, Holoviz Panel and Docker.
3. Implement recording of all pipeline artifacts and recording of the filtered dataframe as the final step of ETL (Extract, Transform, Load) of the pipeline.
4. Automatically use the filtered dataframe in the model, regardless of the DataQC of the pipeline.

Among the steps taken to accomplish this work:

1. Anomaly and Data Drift detection in DataQC and Data Drift pipelines, respectively
2. Pipeline management using MlFlow Tracking

Anomaly detection helps us to assess the cleanliness and quality of our data. Let us consider from the point of view of the application of these data. The principle is that the cleaner and better our data, the better our prediction.

It is important for the model not to have anomalous outliers because it confuses the model. Also, it is important to have consistent data without feature distribution changes.

For instance, if the model has learned some feature corresponds to certain qualities, and the relationship between the feature and its meaning changes, the model can not conclude what is happening. We have to catch such cases and automate their detection, which is the second point in the research list above.

Another example is catching an anomaly during model training. Let us imagine feature distribution, and if one has an anomalous record, it shifts and changes the shape of the original data. The model can no longer understand data limits because of a few anomalous records that bring no information gain.

Machine learning management in MlFlow helps us to keep track of the results of experiments and always be sure how our actions have influenced the experiment result. We can always empirically and repeatedly derive any saved experiment and either repeat it or refine it. Also, the tool provides a good visual representation of metrics, which cannot be a disadvantage. We can also save the model itself in MlFlow, which we can reuse in another experiment or automate.

This solution will allow us to visualize all the steps performed by the data validation pipeline, allowing other developers to view the result of its work without knowing the nuances of its implementation and without wasting extra time. We unified the solution with MlFlow and Docker.

Also, our MLOps architecture allows us to keep track of data trend changes. Consequently, ensure that the model will retain its predictive efficiency over time.

References

1. What You Need To Know About Telepresence Robots: What They Are and Use Cases // [Electronic resource] OhmniLabs Writer. – 2021. - Access mode: https://ohmnilabs.com/content/what-to-know-about-remote-telepresence-robot/
2. Hancock E. Pattern Recognition // [Electronic resource] Journal Pre-proof, Vol. 123 – 2021 - Access mode: http://csitjournal.khmnu.edu.ua/
3. Hwang S., Wug Oh S., Kim S. J. Single-shot Path Integrated Panoptic Segmentation // [Electronic resource] Computer Vision and Pattern Recognition. – 2020. – Access mode: https://arxiv.org/abs/2012.01632
4. He K., Gkioxari G., Dollar P., Girshick R. Mask R-CNN // [Electronic resource] .- 2018. – pp. 1-17. - Access mode: https://arxiv.org/pdf/1703.06870.pdf
5. Girshick R. Fast R-CNN // [Electronic resource] arXiv e-prints. – 2015. – Access mode: https://arxiv.org/pdf/1504.08083.pdf
6. Min Read J. S. An Introduction to the COCO Dataset // [Electronic resource] Roboflow Blog. - 2020. - P. 17. – Access mode: https://blog.roboflow.com/coco-dataset/
7. Amazon.com: Brookstone Rover 2.0 App-Controlled Wireless Spy Tank: Toys & Games // [Electronic resource] Amazon.com. – 2020. - P. 1. – Access mode: https://www.amazon.com/Brookstone-Rover-App-Controlled-Wireless-Tank/dp/B0093285XK
8. Double Robotics - Telepresence Robot for Telecommuters // [Electronic resource] Double Robotics – 2021. - P. 2. – Access mode: https://www.doublerobotics.com/double2.html
9. Beam // [Electronic resource] Beam. – 2021. - P. 1. – Access mode : https://suitabletech.com/products/beam.
10. Amazon.com: Appbot Riley Home Safety Movable Camera Robot: Camera & Photo // [Electronic resource] Amazon.com – 2021. - P. 1. – Access mode: https://www.amazon.com/Appbot-Riley-Controlled-Movable-Safety/dp/B01LWXF28H.
11. Gandhi R. R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms // [Electronic resource] Toward data sience. – 2018. – Access mode: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
12. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection // [Electronic resource] arXiv e-prints. – 2016. – Access mode: https://arxiv.org/pdf/1506.02640v5.pdf
13. Freeze Tensorflow models and serve on web // [Electronic resource] CV-Tricks.com – 2017. - P. 1. – Access mode : https://cv-tricks.com/how-to/freeze-tensorflow-models/ .
14. Shiledarbaxi N. Guide to Panoptic Segmentation +A Semantic + Instance Segmentation Approach // [Electronic resource] Analytics India Magazine – 2021. – Access mode: https://analyticsindiamag.com/guide-to-panoptic-segmentation-a-semantic-instance-segmentation-approach/ .
15. Hui J. SSD object detection: Single Shot MultiBox Detector for real-time processing // [Electronic resource] Medium – 2020. - P. 1. – Access mode: https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06.
16. Choudhury A. Top 8 Algorithms For Object Detection One Must Know // [Electronic resource] Analytics India Magazine – 2020. - P. 1. – Access mode: https://analyticsindiamag.com/top-8-algorithms-for-object-detection/.
17. Hui J. Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and…) // [Electronic resource] Medium – 2020. P. 1. – Access mode: https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359.
18. Evidently AI; Data Drift Report [Official site of EvidentlyAI]. Retrieved from https://docs.evidentlyai.com/reports/data-drift [in English]
19. TensorFlow Data Validation [Official documentation of Tensorflow]. Retrieved from https://www.tensorflow.org/tfx/data_validation/get_started [in English]
20. Holoviz Panel [Official site of Holoviz]. Retrieved from https://panel.holoviz.org [in English]
21. Amazon SageMaker [Official documentation of Amazon SageMaker]. Retrieved from https://docs.aws.amazon.com/sagemaker/index.html
22. GoEmotions [Official documentation of dataset]. Retrieved from https://ai.googleblog.com/2021/10/goemotions-dataset-for-fine-grained.html
23. MlFlow [Official site of MlFlow]. Retrieved from https://www.mlflow.org/docs/latest/tracking.html
24. Boyko N., Kovalchuk R. Anomaly Detection, Data Drift Detection for Time-Series on Dismissal Prediction System – 2022
25. Sculley D., Holt. G, Golovin D., Davydov E., Phillips T., Ebner D., Chaudhary V., Young M., Crespo J., Dennison. D. Hidden Technical Debt in Machine Learning Systems - 2020

Nataliya Boyko **Наталія Бойко**	Ph.D., Associated Professor at the Department of Artificial Intelligence, Lviv Polytechnic National University, Lviv, Ukraine e-mail: Nataliya.I.Boyko@lpnu.ua https://orcid.org/0000-0002-6962-9363	Доцент кафедри системи штучного інтелекту Національного університету "Львівська політехніка"
Roman Kovalchuk **Роман Ковальчук**	Student at the Department of Artificial Intelligence, Lviv Polytechnic National University, Lviv, Ukraine Middle Data Scientist at SoftServe, Data Science, MLOps, Python, Azure, GCP | Microsoft Certified: Azure AI Fundamentals e-mail: roman.kovalchuk.knm.2019@lpnu.ua https://orcid.org/0000-0001-9039-125X	Студент кафедри системи штучного інтелекту Національного університету "Львівська політехніка"